Git

Need of Git:

==========

Git is a free and open-source version control system used to track changes in source code during software development. It helps developers collaborate, manage code history, and work on different features without affecting the main project.

Advantages of Git:

==========

1.Distributed System : Every developer has the full history of the project locally, which allows offline work and fast operations.

2.Speed : Most operations (commits, diffs, merges) are done locally, so they're super fast.

3.Branching and Merging : Lightweight and cheap branches make it easy to experiment, isolate features, and collaborate.

4.Staging Area : You can prepare commits in parts using the staging area, which gives flexibility in how changes are committed.

Disadvantages of Git:

==========

1.Steep Learning Curve : Git commands can be confusing at first (e.g., rebase, merge, reset, cherry-pick). And Understanding its internal model (commits, trees, refs) takes time.

2.Complex Merge Conflicts : When multiple people change the same parts of code, resolving conflicts can be tricky and error-prone.

3.Not Ideal for Large Binary Files: Git isn't optimized for binary files (like images, videos). Even with Git LFS (Large File Storage), performance can still lag.

4.Confusing History Rewrite : Commands like rebase, reset, and filter-branch can be dangerous if misused, especially in shared repositories.

How to Download :

==========

Open thi url : https://git-scm.com/downloads/win after download git

Open command prompt enter >git then you will get git info it means its installed successfully

What is the Working Directory?

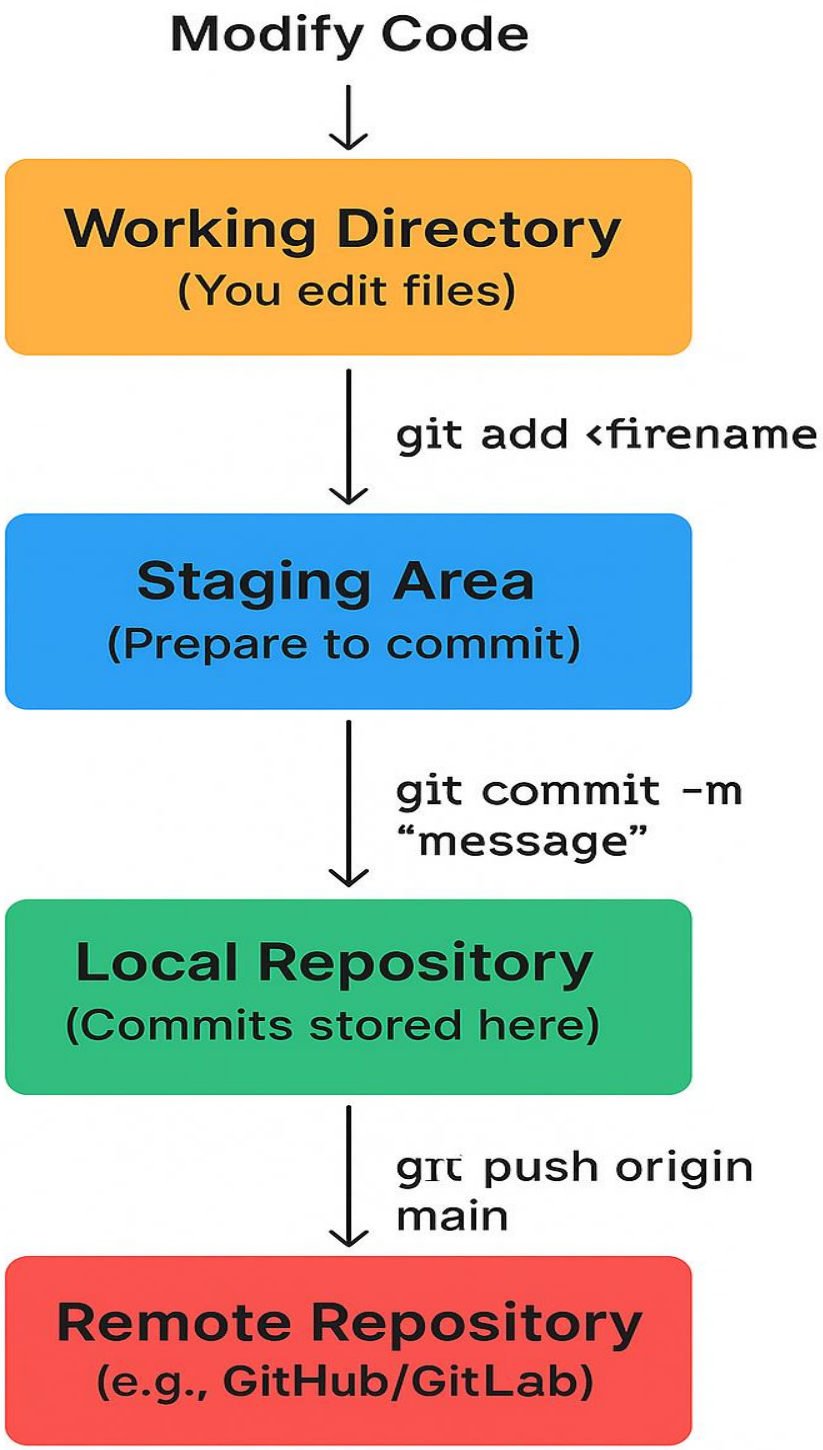
==========

The **working directory** (or working tree) is the **actual folder on your computer** where your project files live — the files you can open, edit, and see in your code editor.

Working Directory = Your project folder where you write and edit code.

Create a project in Eclipse and go to the project path, and open cmd prompt. Then enter

- ➔ **git init ,** after enter
- ➔ **git status,** the files are there in red color, it means those files there still in the working directory.


Modify Code
Working Directory
(You edit files)
git add <firename
Staging Area
(Prepare to commit)
git commit -m
“message”
Local Repository
(Commits stored here)
git push origin
main
Remote Repository
(e.g., GitHub/GitLab)

What is the Staging Area?

==========

The staging area is like a waiting room where you put the changes you want to include in your next commit.
Staging Area = A place where you prepare your changes before saving them permanently (with a commit).

Open your command prompt in above mentioned project workspace and open the command prompt and enter

- ➔ **git add <project name> or <file name>**, then it will enter into Staging Area. We can see the green color files.

What is the Local Repository in Git?

==========

The **local repository** is where Git **saves your commits** — the full history of your project — **on your computer**.

Local Repository = The hidden .git folder that stores all your commits and project history locally.

Open your previous command prompt and enter

- ➔ **git commit -m "**code committed for Demo project code changes."

Here, -m stands for message — we use it to describe the purpose of the commit.

Account Creation :

==========

Click on this link : GitHub · Build and ship software on a single, collaborative platform · GitHub

Follow the steps below:-

==========

Step 1: Download it.
Step 2: Open your folder(which you want to commit to git). In the address bar, enter **cmd.**
Step 3: Enter the **git init**
Step 4: Enter **git status,** and add all files into git staging (**git add .**), commit the changes.
Step 5: git commit -m "Message description**."**

What is the Remote Repository in Git?

==========

The **remote repository** is a version of your project that lives on a **remote server** — usually on **GitHub**, **GitLab**, or **Bitbucket**.

Step 6: Add GitHub remote

Replace <your-repo-url> with the actual HTTPS URL of your repo. In your case, it should be:

- ➔ **git remote add origin** https://github.com/naveen-angati/LogicalPrograms.git

Step7: Push your code to GitHub : Now push your code to the remote origin

- ➔ **git push -u origin master**

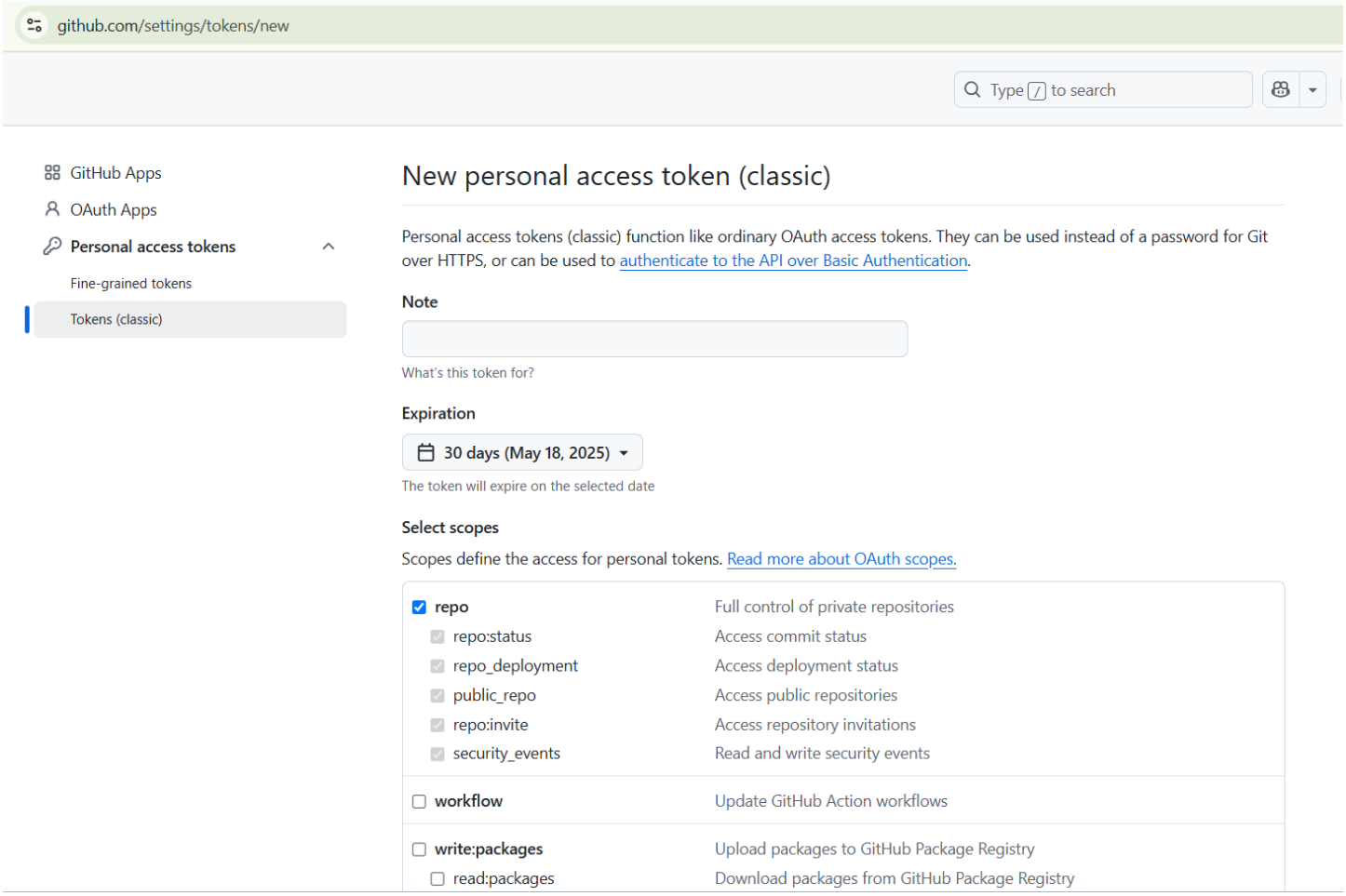
- **push** : Sends your local commits to the remote repository

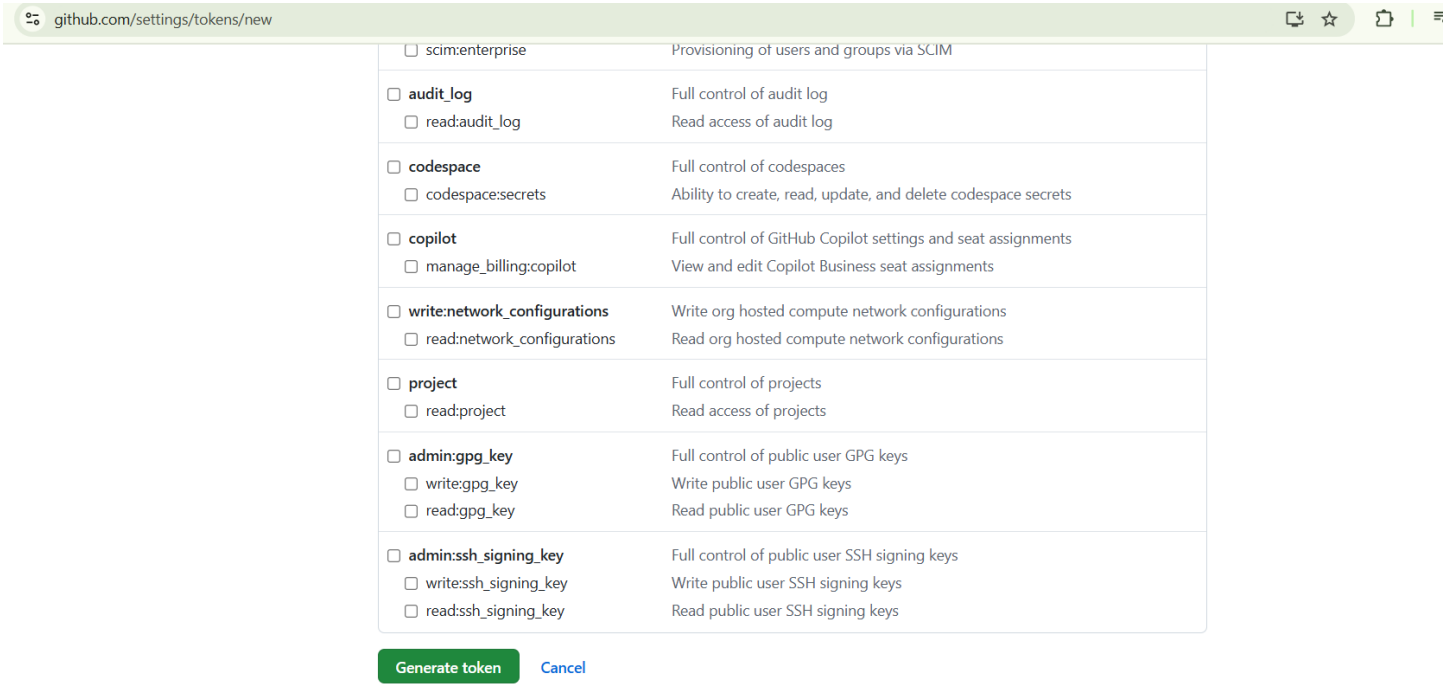
- **-u** : Sets origin master as the default push target
- **Origin :** The remote name
- **Master:** The branch name

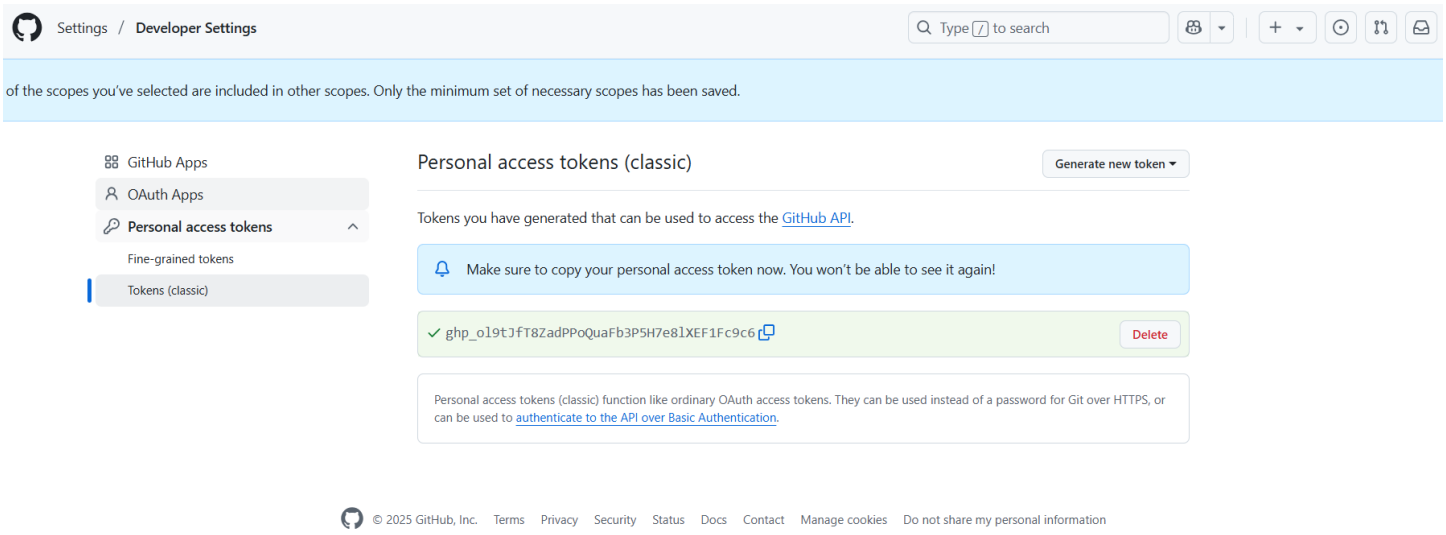
User Name : your username

Password : we have to generate token then we have to past and anter here

https://github.com/settings/tokens







Paste in **Password** section : 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

GIT & GITHUB COMMANDS CHEAT SHEET

==========

Git Basic Setup

git config --global user.name "Your Name" # Set your Git username
git config --global user.email "you@example.com" # Set your Git email
git config --global credential.helper manager # Save credentials for future GitHub use
git config --list # Show all Git config settings

🗀 Git Repository Initialization

==========

git init # Create a new local Git repository
git clone <url> # Copy (clone) a remote repository to your computer

🗁 Staging & Committing Changes

==========

git status # See the current status of files (modified, staged, etc.)
git add <file> # Stage a specific file for commit
git add . # Stage ALL files in the folder
git reset # Unstage all staged files (but keep changes)
git commit -m "message" # Commit staged changes with a message

🌐 Remote Repository (GitHub)

==========

git remote add origin <url> # Link your local repo to a GitHub repo
git remote -v # Show linked remote repositories
git push -u origin master # Push your code to GitHub (master branch)
git push -u origin main # Push your code to GitHub (main branch)
git pull # Fetch and merge changes from GitHub to your local repo
git fetch # Fetch changes from GitHub (without merging)

🔁 Branching & Switching

==========

git branch # List all branches in your project
git branch <name> # Create a new branch
git checkout <branch> # Switch to another branch
git checkout -b <name> # Create and switch to a new branch
git switch <branch> # Switch to an existing branch (modern alternative)
git switch -c <name> # Create and switch to a new branch (modern alternative)
git merge <branch> # Merge changes from one branch into the current branch

🗑 Cleanup & Removal

==========

git rm <file> # Remove a file and stage the deletion
git clean -fd # Delete untracked files/folders (use with caution)
git reset # Unstage files from staging area

```
git reset <file>                    # Unstage a specific file
git reset --soft HEAD~1             # Undo last commit (keep changes staged)
git reset --mixed HEAD~1            # Undo last commit (keep changes but unstage)
git reset --hard HEAD~1             # Completely remove the last commit (dangerous!)
```

📜 Logs & History

```
git log                             # View full commit history
git log --oneline                   # View commits in a compact format
git show                            # Show details of a commit (default: last)
git diff                            # Show file differences between commits or changes
```

💣 Undo Commands (Use with Caution)

```
git checkout -- <file>              # Discard local changes to a file
git revert <commit>                 # Create a new commit that undoes a previous one
git reset --hard HEAD~1             # Remove last commit and changes (⚠ dangerous!)
```

🔄 Syncing with GitHub

```
git fetch                           # Get the latest changes from GitHub (no merge)
git pull                            # Get and merge changes from GitHub
git push                            # Upload local commits to GitHub
```